Testing a Neural Network Accelerator on a High-Altitude Balloon

Gilbert Clark
LCN Branch
NASA Glenn Research Center
Cleveland, OH, USA
gilbert.j.clark@nasa.gov

Geoffrey Landis
LEX Branch
NASA Glenn Research Center
Cleveland, OH, USA
geoffrey.landis@nasa.gov

Ethan Barnes
Intern
NASA Glenn Research Center
Cleveland, OH, USA
ethancbarnes@gmail.com

Blake LaFuente
Intern
NASA Glenn Research Center
Cleveland, OH, USA
blakelafuente@gmail.com

Kristina Collins
Intern
NASA Glenn Research Center
Cleveland, OH, USA
kristina.collins@case.edu

***Abstract*— The cognitive communications project has been working to refine artificial intelligence and machine learning approaches to support their deployment and sustained use in space environments. It has historically been difficult to implement such techniques on space platforms, however, due to the computational requirements they levy onto general-purpose avionics hardware. While technologies exist to accelerate the computation of aspects of neural networks, such platforms have not historically been deployed in space environments. Given that testing payloads in such environments can be both cost- and time-prohibitive, high-altitude balloons can be used as a way to approximate a space environment at a much lower cost, thus providing a cost-effective way in which to test newer approaches to hardware acceleration for artificial intelligence which may be deployed onto spacecraft more directly.**

This paper describes a successful test of a commercial off-the-shelf neural network accelerator on a high-altitude balloon. It begins by explaining our selection criteria when evaluating different commercial neural network acceleration techniques: primary considerations include size, weight, and power (SWaP) as well as ease of integration. Next, the paper describes the development and implementation of an experimental flight test platform: flight and ground components are discussed. Afterward, the paper discusses the experimental payload itself: this includes the experimental procedure as well as the specific image and method used for testing. Finally, the paper concludes with an evaluation of both the experimental device tested at altitude as well as the flight test framework itself, identifying how the existing platform can be used to continue testing commercial off-the-shelf (COTS) solutions for acceleration.

Keywords—artificial intelligence, embedded systems, system verification

I. Introduction

One commonly cited source of the "Internet of Things" (IoT) states that it began as the title of a presentation that was intended to link radio-frequency identification technology with the internet at large [1]. Since then, the phrase has evolved into a larger phrase that describes the connectivity between an ever-growing number of distributed sensors, instruments, and embedded devices. While each individual device tends to be low-power, there are a number of approaches that allow such devices to capitalize on more powerful aspects of the networks they are connected to.

Data offload to the cloud offers one approach to augmenting the capabilities available to power- and compute-constrained IoT devices: there has been a substantial amount of work in developing cellular (and other) approaches to communications that have been tuned for IoT applications. These approaches focus on allowing sensors to offload their data to specialized services running, for example, on the Amazon Web Services, Microsoft Azure, or other internet-based platforms. Such platforms often provide aggregation and analysis services, which are then capable of applying increased amounts of compute power in a datacenter toward analyzing, repackaging, and presenting data to users. The only requirement this levies onto the IoT device is an expectation of connectivity: that it will be able to offload its data for this processing to take place. This approach shares a number of common characteristics with the way science data is processed by spacecraft today.

A. *Space Assets as IoT Devices*

Space assets often act as extremely expensive wrappers for an (also expensive) instrument. Fundamentally, the intent of the architectural approach to space research is similar to the architecture often observed in common IoT applications: find a way for the data gathered by an instrument on a device to reach the people (*e.g.* scientists and engineers) who are interested in reviewing it. Spacecraft, like most IoT devices, tend to be constrained in terms of their size, weight, and power: when designing a space mission to last for years, every milliwatt counts. While it may be easier to replace a node taking measurements in a forest than it is to replace a node taking measurements in space, in neither case does one want to venture out to replace a battery more often than is absolutely necessary.

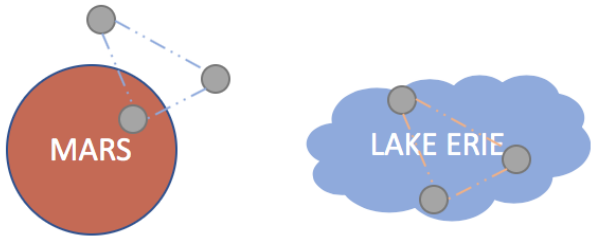

Figure 1 - Sensor Networks in Space vs. Sensor Networks on Earth

Generally, this means that compute and RF capabilities are planned extremely carefully for flight systems. In most instances, the model adopted by the mission resembles that of the model adopted by modern IoT cloud applications: virtually no processing happens on the spacecraft, and the supporting systems on the spacecraft are simply intermediary systems to support buffering data and returning it to the ground. This approach is a straightforward one, but does present some difficulties not commonly associated with IoT devices.

B. Constraints on Spacecraft Communications

Traditional space networks operate more like circuit-switched phone systems than like packet-switched terrestrial networks. There are a number of reasons for this, but many of them trace back to the physical mechanics involved with closing and maintaining an RF link. Satellites in non-geostationary orbits move into and out of range of ground stations relatively quickly, leading to dynamic link characteristics that can vary wildly between successive passes. Further, ground stations are normally tuned to optimize for maximal gain, leading to highly directional RF systems that can speak with a relatively limited number of assets in parallel. Additionally, such high-gain equipment is relatively expensive to operate and maintain: even if the capital expense of building a new ground station can be supported at a given moment in time, the operational expenses represent a long-term infrastructure commitment. Thus, communication with the ground can often require a nontrivial degree of human intervention to complete and maintain.

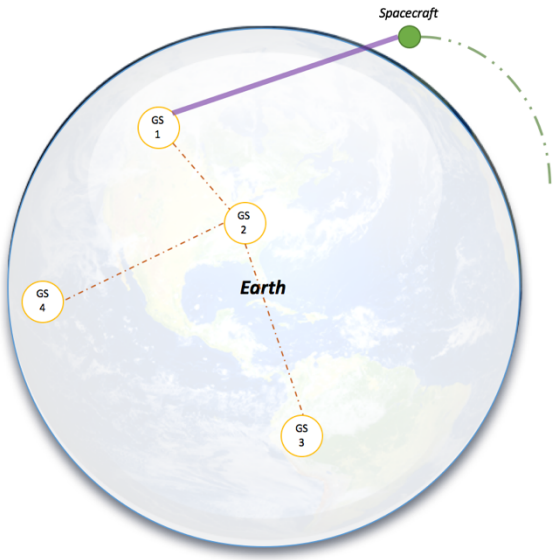

Figure 2 – Intermittent Contact with Networked Ground Stations

The orbital mechanics can be simplified (to an extent) through the use of assets such as geosynchronous relay satellites. Unfortunately, while geosynchronous relay spacecraft can be an option for larger missions, they are generally a challenge for smaller missions to effectively utilize. A satellite in geosynchronous orbit is over 22,000 miles above mean sea level, whereas satellites in low-Earth orbit are generally less than 1,250 miles above the surface: as such, it requires substantially more power to close a link with a relay satellite than it does to close an equivalent link directly with a station on the ground. Addressing the power and thermal considerations necessary to facilitate the use of a geosynchronous relay at a nontrivial bit rate normally promotes a spacecraft out of the "small" category.

C. Toward Computing at the Edge … of Space

While there are relatively hard physical limitations on RF links between spacecraft, limitations on compute aspects of modern spacecraft design are somewhat more fluid. Innovations in architecture and manufacturing process allow modern CPUs to iteratively improve on processing capabilities without affecting the resulting size or weight of the processing package. Further, various advances in hardware acceleration for specific algorithms and approaches (*e.g.* neuromorphic processors) can offer savings when compared to general-purpose CPUs that would otherwise be necessary to accomplish an equivalent task [4, 13].

These improvements have led to a marked interest in the idea of running nontrivial compute tasks directly on spacecraft before data passes through the link. This interest mirrors, for example, the interest in "edge computing" (or, more recently, "fog computing") on terrestrial IoT systems [2]. More efficient and powerful CPUs have facilitated more powerful processing capabilities on individual IoT systems. By running such processing tasks before the data leaves a system (be it space or terrestrial), one can significantly reduce the data requirements for a system while simultaneously improving *e.g.* the speed at which a system can analyze and react to a dynamic environment.

D. Technology Readiness and the High Altitude Balloon

While the idea of edge computing sounds attractive in theory, mission designers tend to be extremely hesitant to adopt unnecessary risk in systems which are not required for a successful mission. This leads to something of a proverbial chicken-and-egg problem with implementing new technologies in missions: a technology cannot be proven if it does not fly, but no one wants to assume the risk inherent to flying an unproven technology. While missions are commissioned entirely to demonstrate unproven communication technologies, such missions are themselves somewhat rare and carefully structured in terms of their lifetime and the types of experiments they can support. Further, in order to be considered for such a mission, a technology must reach an acceptable level of development. To better characterize the stage of development for a specific technology, NASA adopts what it refers to as a technology readiness level (TRL). The TRL scale generally ranges from 1 to 9, with 1 being a largely abstract idea, 5 being something that has been prototyped on a breadboard / emulated environment, and 9 being something that is battle-tested and has flight heritage [14]. There are requirements for a technology to progress through the various stages of development, one of which involves applied demonstrations in flight-like situations.

A high-altitude balloon (HAB) experiment presents one way to take a technology at a lower TRL and demonstrate that it can operate in space-like conditions for a specific period of time. HAB missions are especially attractive in many cases because they are cost-effective and simple: one can combine a parachute, a small foam container, and a weather balloon, and find themselves with a functional way to validate successful operation of an experimental payload at altitudes of 25 - 30 km above sea level, an altitude at which it will be above 99% of the Earth's atmosphere, for a cost of $1,000 USD or less. While the conditions found at 25 – 30 km are

not identical to those commonly found in low-earth orbit, many challenges are similar: at such high altitudes, for example, convection cooling is less effective. The ambient air temperature is also low enough that devices can begin to operate incorrectly, and various aspects of design (*e.g.* battery capacity [6]) need to be adjusted to suit the environment.

II. Experimental Design

A. Experiment Overview

For reasons described above, we elected to use a HAB to demonstrate the operation of a commercial off-the-shelf (COTS) neural network accelerator in space-like conditions. The intent of this experiment was to elevate the TRL of the concept of on-board processing in the abstract. Since this was an early experiment in this area, and the experiment was intended to be largely student-led, we elected to focus on three high-level objectives for our experiment: simplicity, low size, weight, and power (SWaP), and hardware neural network support. We also were incidentally constrained by cost and availability of parts.

Simplicity was a must because there were only 10 weeks in which to construct, test, validate, fly, and retrieve an experimental payload. No mechanical components of the balloon were specifically constructed for flight: the payload was contained in a simple polystyrene box. Generally, components were over-provisioned and relied on Arduino-compatible libraries to ease the software development process. The neural network accelerator was similarly envisioned as something that would be straightforward for students to work with.

Minimal SWaP was an objective for many reasons, each of them important to the success of the intended flight. First, minimal SWaP was desired because flight assets are necessarily limited in the amount of size, weight, and power available to them. With this in mind, the lower the cost of the computational and communication elements, the more space might be available for science on the asset. Secondly, to ease the validation process, the payload was constructed to be sufficiently lightweight that it would not fall within the applicability parameters of 14 C.F.R. §101.1. Specifically, these regulations placed a hard limit of four pounds (1.8 kg) on the project.

Hardware neural network acceleration support was a requirement because it was a core aspect of our envisioned edge compute platform. While neural networks have been implemented on small microcontrollers in the past [3], such platforms place severe constraints on the format, size, and function of such networks. Newer hardware acceleration platforms, on the other hand, allow massive neural networks to operate in the same SWaP envelope as that of a microcontroller: IBM's TrueNorth platform offers 1 million neurons and 256 million synapses at 65 mW of power consumption [4]. Field Programmable Gate Arrays (FPGAs) are another alternative for acceleration of specific types of neural network compute workloads: while their performance per watt does not tend to reach the same levels of that found within dedicated ASICs [5], they do still represent an improvement over that of implementations based entirely within general-purpose CPUs.

Finally, our experiment had a budget of approximately $300 USD for parts and expenses. This amount included only parts that we intended to purchase for the experiment: many elements were recycled from similar applied projects and low-cost experiments that had been run in the past. Thus, when considering purchases, we first considered whether the parts we had available already would be adequate to perform the intended task.

With those objectives in mind, we proceeded to the design phase of the project. After some consideration, the design was split into two major subsystems: an infrastructure subsystem and an experiment subsystem. The infrastructure piece supported *e.g.* power distribution and real-time communication with the ground, while the experiment piece consisted of a neural network accelerator (as well as any supporting hardware / software that might accompany it). While not covered here in depth, the infrastructure element also had a requirement to interface with a secondary payload that also resided upon the same physical balloon.

B. Communications Infrastructure

The infrastructure of the balloon included four distinct parts: the power system, a microcontroller, a radio, and various flight-related recording instruments. Each of these elements will be covered separately in the following sections. Note that a graphical overview of the balloon's wiring and systems may be observed in Figure 3 (below).

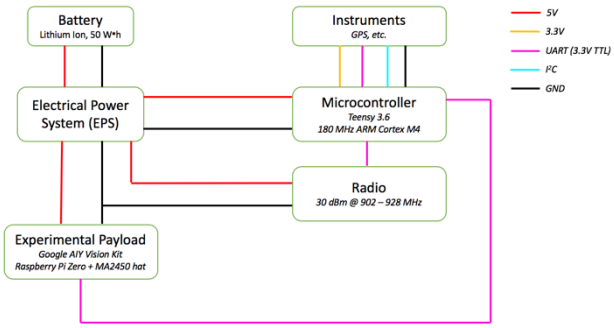

Figure 3 - Infrastructure Overview

1) Microcontroller

When identifying a microcontroller, there was a desire to obtain a chip that included support for a wide variety of low-speed serial inputs and outputs. The chip was also intended to be simple to program and work with, lightweight, and power efficient, while simultaneously offering sufficient resources that relatively little time would need to be spent on *e.g.* ensuring code could fit into available flash and RAM on the device. A number of development boards were evaluated for this purpose.

The Arduino UNO was the first such device: it was attractive due to its widespread adoption and simple programming interface, but was ultimately unable to be used here due to the low number of UARTs available on the part. There was a requirement to support four simultaneous UARTs on the balloon, but the Arduino has hardware support for only one. We were not confident in this device's ability to emulate three other UARTs in software while simultaneously interacting with other instruments over I^2C and logging data to an SD card – this may have been within the realm of possibility, but would have required some degree of optimization and careful implementation to realize given the resources available.

The second and third options considered were ARM Cortex M4 devices from STM32. These parts supported more

I/O than the Arduino and had reasonable power requirements, but were more complex to program: rather than relying on the well-known Arduino set of functions, the STM32 development boards exposed the function of the chips and devices directly. This ensured they would remain an option, but something Arduino-compatible was preferred.

One such board (and the final candidate evaluated here) was the Teensy 3.6 development board. This board included an ARM Cortex M4 processor (nominally clocked at 180 MHz), but offered drop-in compatibility with the Arduino programming toolchains and approach. It included a number of I/O capabilities, and also included software and hardware support for SD cards directly on the board: this removed the need to interface with an external device to obtain support for *e.g.* logging instrument data to SD cards during flight. For these reasons, the Teensy 3.6 was selected.

2) Radio System

The radio system, in this instance, was constructed from a pair of radios that operated within the unlicensed 902 – 928 MHz band. These radios were selected primarily because they had been used in previous experiments with some degree of success, so the acquisition of a new system would not be needed. With that said, the radios used for this project could internally support transmission at up to 30 dBm, and were reportedly able to support a 10 Kbps link down to -110 dBm. Thus, the radio system provided a maximum link budget of 140 dB. Given that the balloon was expected to reach an altitude of 25 – 30 km, calculations indicated that a link would remain available through much of the flight and that there would be a reasonable degree of margin to account for interference and miscellaneous losses (*e.g.* antenna polarization mismatch, impedance mismatch, and multi-path losses).

Generally speaking, the radio's high supported transmission power meant that it was the device that drew the most power. From the associated user manual for the device, transmission at 30 dBm was expected to draw 710 mA at 5V (or 3.6W). As a result, the duty cycle for the radio was minimized to avoid issues with overheating and to maximize the life of the battery.

3) Power System

For systems that operate at high altitudes and low temperatures, environmental conditions can have a negative impact on battery life and performance [6, 7]. When the power system was designed, it was built with an objective to support roughly 9 hours of continuous operation: this accounted for a flight of a few hours, followed by a retrieval process that might take longer to complete. Communication with the balloon can be useful during the retrieval process, so the battery was over-provisioned to support such a use-case.

In this case, the battery selected was a 5V, 10,000 mAh lithium-ion device designed to deliver power exclusively through a pair of USB ports. These USB ports presented a convenient way to power most common system-on-chip devices without necessitating direct wiring. For common chipsets and boards (*e.g.* the Raspberry Pi Zero), this removed the need to bring power to header pins directly, reducing the chances that *e.g.* a mis-wired connection would damage a component. For cases where more direct power was required, this was accomplished by breaking one USB port out into a 5V rail and a ground rail on a power distribution board.

The power distribution board was a small breadboard that offered a 5V rail, a ground rail, a 3.3V rail, and voltage level translation. For simplicity's sake, the system diagram has been simplified to not route the level translation and 3.3V through the distribution board: this does not sacrifice accuracy because the 3.3V rail involved was sourced from the microcontroller, and because the level translation was not tied to any specific rail.

4) Instruments

Data gathering and recording was performed by independent sensors included on the balloon. An inertial measurement unit was used to observe the orientation and movement of the payload during flight. These measurements were recorded with the intent to understand the kind of stresses that were observed during the flight. Additionally, the IMU data was expected to offer insight with respect to how the antenna moved during flight.

A GPS module was used to track latitude, longitude, and altitude of the balloon. Data retrieved from this module was included in telemetry frames and downlinked from the flight system on a regular basis. These frames were fed into a real-time recording and data visualization system on the ground, which in turn allowed the balloon to be tracked in real-time throughout its flight.

5) Software and Ground Infrastructure

The software development effort was split into two distinct pieces: ground and flight. The flight software was written in the C++ programming language and relied on the Arduino library to support data communications. On flight, the intent was for the microcontroller to manage data gathering and manipulation, as well as push telemetry and payload data to the radio module for downlink. The microcontroller also responded to link characterization messages that allowed the ground software to *e.g.* record message latency and to indicate the signal strength as measured on the ground.

Figure 4 - Ground System

The core of the ground software was written in C++. The ground software was written to record and visualize information received from the balloon in real-time (*e.g.* GPS data, signal strength, ambient temperature, ambient pressure, etc). Data received on the ground was fed into an indexing backend called ElasticSearch, and visualized through a web browser: the visualization was constructed through a software package called Kibana.

All ground software was run on a Raspberry Pi 3 system (pictured in Figure 4 above). This system was configured to operate as an 802.11 access point: to view data from the balloon flight, one would first join the 802.11 network, and then access the web interface running on that access point directly. Relevant flight data was also displayed directly on a small LCD. This configuration allowed multiple users to view flight data as the experiment progressed. Communication with the radio system was supported through a standard USB to RS-232 adapter.

C. Experimental Payload

Once the infrastructure had been designed and largely implemented, candidate neural network acceleration devices were identified. Solutions evaluated included those based on FPGAs as well as solutions based on low-cost hardware. Note that this trade was not intended to act as an exhaustive exploration of the space: instead, it was to select a device that was approachable enough and simple enough that an experiment could be designed and implemented within the time available to students in which to work.

1) Neural Network Acceleration Approaches

Generally, approaches to accelerating neural networks fall into a few different categories. The first category is that of GPU-accelerated computing, which relies on the massively parallel nature of modern graphics cards to accelerate the computations associated with both training and executing neural networks. GPU acceleration is an attractive general-purpose solution, and is not limited to execution on larger platforms: the Jetson TX2 platform, for example, supports 256 CUDA cores within a power envelope of 15W [9].

Another broad category describes FPGA-based approaches to acceleration. There are generally multiple ways to leverage FPGAs to accelerate such computation: one approach is to utilize the FPGA as an OpenCL target and develop kernels for that directly [10]. In this manner, the FPGA acts in a manner similar to that of a GPU, where it accelerates specific aspects of general neural network computation. A second, more specialized approach to FPGA-based neural network acceleration involves building neural networks that map to the FPGA architecture more directly [11]. This has the advantage of offering substantial improvements to performance, but one disadvantage is that different approaches to such acceleration often necessitate specific interfaces. Such approaches also necessarily limit the use of an FPGA to the specific type of problem for which an optimized solution has been devised.

A third category involves the use of more specialized hardware ASICs to support accelerated computation. Such ASICs can range from specialized arrays of vector processors that target specific applications (such as image processing) [12] to more architecturally unique approaches that support generalized neural network implementations [4, 13].

2) Evaluating Acceleration Options

Given the dizzying array of solutions supported, some initial selections were made to limit the scope of the evaluation. First, more esoteric solutions based on unique hardware were eliminated: given that there were cost constraints placed on the balloon and that access to such hardware tends to be somewhat limited, it was not deemed to be a practicable approach to acceleration. Further, such hardware normally necessitates the acquisition of specialized toolchains to effectively support development, requiring some time to be invested in a specific solution before any gains could be realized.

Next, the field of candidate devices was again reduced in scope to focus on devices that supported the well-known TensorFlow and PyTorch libraries. These libraries accelerate the development and utilization of approaches to artificial intelligence. Use of these libraries would avoid specific one-off solutions that, while efficient, would very likely be outside the realm of feasible implementation given the resources and timeframe available for development. It would also allow students to rapidly test and experiment with a number of existing Artificial Intelligence (AI) and Machine Learning (ML) approaches that had already been developed based on these frameworks, which was perceived as a win.

The field was narrowed further based on an evaluation of the backgrounds of the individuals performing the experiment. The team involved had limited FPGA experience and, given the compressed timeframe of the experiment, also had limited time available to them in which to learn. As such, while FPGAs should be evaluated more closely in the future, they were not deemed to be a good fit for the purposes of this particular experiment.

The Jetson TX2 was also removed from consideration. One factor that influenced this decision was the relative power cost: while the power utilization can run around 7.5W in specific modes, the power utilization was still significantly larger than that of other devices on the balloon. More important, however, was that there was no guarantee the balloon would be recovered. Thus, while the $599 MSRP for this board meant that it would be replaceable, it would not a device that would be comfortable to lose. Since the experiment was more about demonstrating the concept than about pushing the boundaries of embedded computation, it was determined that another device might be better suited to demonstrate this concept in the abstract.

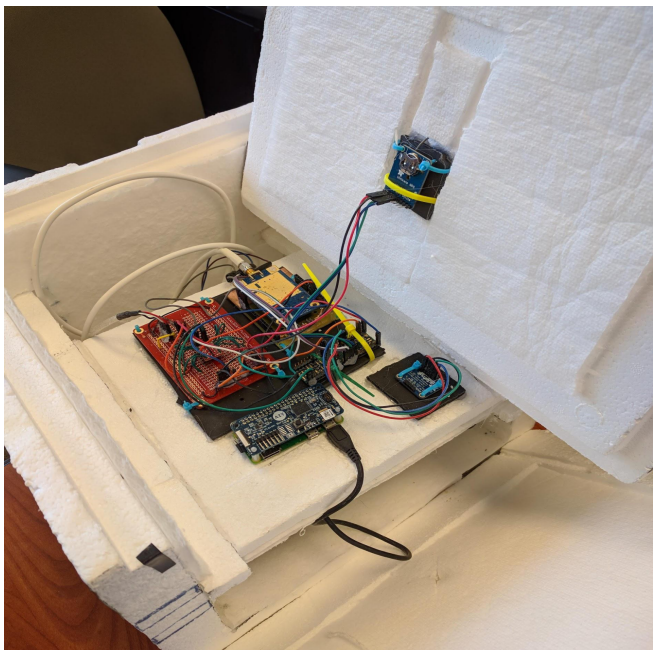

Figure 5 – Flight Infrastructure / Payload Assembly

After further deliberation, candidates were down-selected to a pair of acceleration devices that were both based on the Movidius Myriad 2 chip. The Movidius Myriad 2

incorporates an array of 12 128-bit SIMD vector processors (clocked at 600 MHz), onboard memory and an associated memory fabric, hardware accelerators for specific image transformation and processing tasks, and a lightweight RISC processor [12]. Documentation for the Myriad 2 chipset lists power consumption as less than 500 mW in most cases. The device allows the user to export certain networks from TensorFlow and Caffe and execute them directly on the chip, removing the need for the general-purpose processor (in the case of the Raspberry Pi Zero, the single 1 GHz BCM2835) to perform those tasks.

One of the candidate devices was a Neural Compute Stick: this is a USB stick that incorporates the chip, and can be plugged into many devices for use. The other was a chip (the MA2450) that had been integrated into an extension board (called a "hat") for a Raspberry Pi Zero. A complete kit (called a Google AIY Vision Kit, consisting of a camera, Raspberry Pi Zero, SD card, cardboard assembly, and hat) was available for $89.99 MSRP. The kit also included a ready-made environment (on the SD card) that came with a series of example applications that could demonstrate use of the MA2450 included on the hat.

Overall, the platform appeared to draw an acceptable level of power given available battery constraints, was cheap enough that risk associated with its loss would be small, utilized minimal size and weight, and supported rapid design and development of experiments that would allow students to experiment with the chip. As such, this was the device that was selected to be included in this experiment.

III. Experiment, Flight, and Results

A. Experimental Design

Once an appropriate hardware platform had been identified, the next question was how the platform could be used. Upon conversation, it was determined that one useful application of a balloon would be to validate the device's correct operation at altitude. The intent was to provide the chip with a task that would be representative of an actual task it might perform. Timing information and results associated with the execution of this task would be recorded throughout the flight, and those results would be evaluated upon successful recovery of the payload.

For the purposes of identifying a representative workload, the students identified an image classification routine. The students next identified an image that could be classified – the specific image analyzed will not be reproduced here to avoid any potential issues with copyright. Once the image and routine had been identified, students constructed a script that would repeatedly classify the image. The time taken to perform each individual classification was recorded, as were the results of the classification. Classifications were repeated as quickly as the platform would allow.

The experiment was first performed for an hour on the ground to baseline the expected results. Results were reviewed to ensure that they were, in fact, deterministic when run repeatedly. Reported values were also analyzed to determine how much time the routine was expected to take in normal circumstances. Once this process had been completed, the device was added to the balloon and connected to the EPS in preparation for flight.

B. Assembly and Flight Planning

Glenn Research Center lies next to Cleveland-Hopkins International Airport. As such, airspace at and around the center is restricted, and not well-suited to supporting a launch. The center's proximity to the lake also made it very likely that, should approval to launch be gained, the resulting landing site would fall somewhere in Lake Erie. As such, a launch site was selected at a location around 100 miles southwest of Cleveland. Projections (based on weather conditions and balloon data) indicated that the balloon would travel roughly 30 miles northeast – this put its anticipated flight path and landing site well outside of any restricted airspace. Projections additionally showed that the balloon would generally remain clear of any densely populated areas. Though not technically required, a notice to airmen (NOTAM) was filed in support of the launch.

The final assembly included the communications infrastructure payload, the neural network experimental payload, a secondary experimental payload (not described in this paper), a GPS tracking device, and a small HD camera (to record the balloon's flight). Integration testing between the communications infrastructure and the secondary experimental payload occurred throughout the course of the development process.

The balloon assembly consisted of a high-altitude weather balloon, a parachute, and the payload assembly itself. During the launch, the balloon would steadily rise, and an ever-growing difference in pressure would force the balloon to expand outward. At a pre-determined altitude, the balloon would finally stretch beyond its breaking point and burst. The payload would then begin to fall, and naturally deploy an attached parachute. The payload and parachute were not steered: various models and tools were used to predict the flight path, but no aspect of the flight path was actively controlled during the flight.

Final assembly of the balloon was completed on-site at the launch location. Validation of the assembly occurred after both the communications and experimental payloads had been powered on – a final systems check was completed immediately before launch. Once systems were verified, the balloon was released into the heavens.

C. Balloon Flight

1) Communication System Performance

Initially, communication with the balloon was maintained through an antenna mounted to a 13-foot portable mast. Approximately 10 minutes into the flight, we switched our ground antenna to an omnidirectional antenna mounted to the roof of a vehicle, and began to follow the balloon's recorded GPS locations and projected flight path as it moved northward. A stable communications link was maintained through much of the balloon's flight, allowing for real-time displays of metrics such as signal strength, pressure, and total data transmitted / received through the communications system on the balloon.

2) Flight Metrics and Altitude Modeling

One element that is critical to understanding the characteristics of a balloon's flight is the way its altitude changes over time. To support altitude tracking, the balloon included a pressure sensor. This instrument recorded the ambient pressure (in Pascals) at a rate of one sample every ten seconds.

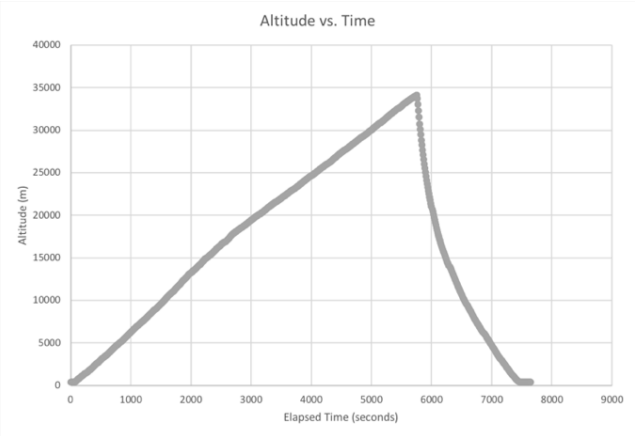

Figure 6 - Balloon Altitude vs. Time

After the flight had concluded, it was necessary to take this raw pressure data and convert it into altitude data. To assist with this process, the National Oceanic and Atmospheric Administration (NOAA) publishes Global Forecast System (GFS) data on a regular basis. This data includes radiosonde data, which maps a number of specific pressures to their corresponding geopotential height. Note that, while geopotential height is generally not equivalent to geometric height, the two are treated as identical for the purposes of this discussion.

To facilitate the conversion from pressure to altitude, specific data points (from July 31st, 2018) were retrieved for a nearby weather station. A model was empirically derived and applied to this data set:

$$46036 + \left(\log\left(\frac{P}{100}\right) * -14712\right) * \left(1 - e^{-8.25*\left(3.013*\log\left(\frac{P}{100}\right)\right)}\right) \quad (1)$$

Equation 1 - Estimating altitude based on recorded pressure

Given a pressure reading in Pascals, this model could be used to estimate the resulting height of the balloon in meters (to an accuracy of within 100m). When this was applied to the pressure data points recorded during the flight, it became possible to compute the instantaneous altitude of the balloon. From these instantaneous altitude values, additional metrics could be calculated (such as the maximum altitude and ascent rate of the balloon).

Based on the recorded pressure readings and the model described above, the balloon remained in the air for a little over two hours. It ascended at a rate of 6 meters per second for 1.6 hours before reaching a peak altitude of 34.1 km (112,000 feet). Once the balloon burst, the payload entered free fall for a moment before the parachute deployed and began to slow the payload's descent.

The payload's landing location was identified within a few hours of landing, but recovering the payload took some additional time: the sun set before the recovery team arrived on site to retrieve the payload. As such, payload recovery was successfully completed the following day.

After review of the data, the payload's 5V / 10,000 mAh battery (50 Wh) supported 15.5 hours of continuous operation. Assuming that all battery capacity was available for the payload to use (which is somewhat conservative), we can thus estimate the average power utilization of the communications and experimental neural network payload at less than or equal to 3.23 W. The experimental configuration did not include a means to differentiate between the power utilization of the communications infrastructure and the experimental neural network payload itself – this is expected to be corrected in future revisions of the platform.

Figure 7 - Image Recorded at High Altitude

D. Experimental Results

The neural network payload performed as expected throughout the duration of the flight. Classification results did not change, and no errors were reported. The classification task was completed close to 6,500 times over the course of the 15.5 hours that the payload was active. During the time the balloon was airborne, the classification task took an average of 8.615 seconds to execute. The execution time did not follow any noticeable trend during the course of the flight.

There was a substantial amount of noise observed in the general execution time. To date, an exact source of this noise has not been determined. However, given that there is only one processor available on the Raspberry Pi Zero, there could be some degree of noise introduced into the results as higher priority tasks are run in addition to the classification experiment run here. Additionally, the Raspberry Pi Zero's storage medium (a microSD card) is a shared resource: if the SD card were *e.g.* in use during times that image was being read or written, this could have acted as a substantial source of error with regard to the recorded runtime for individual executions.

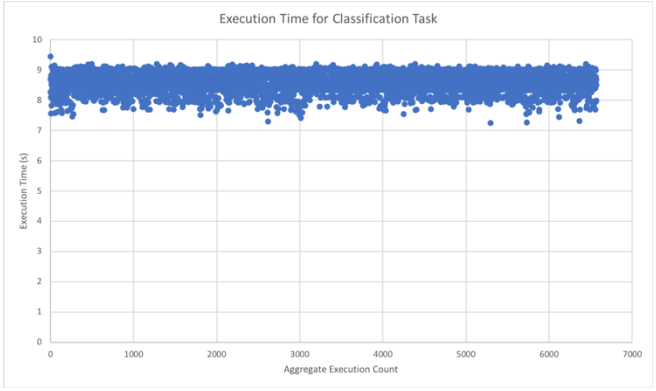

Figure 8 - Execution Time for Classification Task

IV. Conclusions and Future Work

A. Experimental Conclusion

Based on the results observed, it is our conclusion that both the Raspberry Pi Zero and the MA2450 worked as designed: there were no errors logged during executions, and the distribution of classification runtime remained consistent

throughout the course of the 15.5 hours that the experimental payload remained powered on.

Additionally, based on experience with the communications infrastructure during the flight, we determined that the radio and antennas used were sufficient to maintain a stable, bi-directional link with the payload throughout much of its flight.

B. Future Efforts – Communications Infrastructure

Much of the engineering effort that went into the construction of this experiment revolved less around the neural network itself, and more about the test harness (in this case, the supporting hardware and radio system that allowed for communication with the payload(s) under test). However, the communications infrastructure developed in support of this experiment was intended to be reusable, such that it can be used in the future to significantly reduce the non-recurring investment required to build and fly a communications package for a high-altitude balloon.

Further, the communications infrastructure itself offers an avenue for pursuing experiments in the future. By exploiting the ability of neural networks to predict new values based on the way those networks have been trained, it should be possible to use the neural network chip tested here to optimize the performance of the communication system itself. This could, for example, be realized through dynamic adjustments to the transmission power utilized by the balloon, or through decisions to transmit more or less frequently based on a number of parameters. One example of such a parameter might be the amount of data currently awaiting downlink.

C. Future Efforts – Neural Network Accelerator

Moving forward, the neural network platform itself could be explored in more depth. While CPU-based networks could be an option if properly optimized [8], it is expected that the emphasis would remain on hardware acceleration approaches to neural networks. This idea could be realized by testing alternative platforms in similar HAB experiments (*e.g.* FPGA-based approaches to AI) to validate their operation. Alternatively, there is further opportunity to take the neural network accelerator employed for this experiment and test it with arbitrary networks that have been constructed in toolkits such as PyTorch and TensorFlow. Such networks could be applied to optimizing the communications system itself, as indicated earlier, or could be applied toward *e.g.* real-time vision processing of image and instrument data.

D. Conclusion

More generally, the authors hope that these results demonstrate a first step toward more widespread inclusion of AI-centric on-board processing capabilities in future space missions. As such technology matures and finds its way into space, the opportunities for automation and autonomy it brings with it could not only streamline network and spacecraft operations, but more generally serve as a means to push the boundaries of exploration.

In order for such a vision to ever be realized, however, experimentation and adoption of such technologies will be key. It is our hope that this paper might offer an example of how HAB experiments can be used as a low-cost, simple means to demonstrate how specific technologies and concepts can be deployed and used in space-like conditions.

ACKNOWLEDGMENT

The authors wish to express their appreciation to Brian Willis, Steve Hall, David Chelmins, and Robert Jones for their contributions to the construction, validation, and successful flight of this experiment.

The authors wish to express their gratitude to Alan Hylton, Dr. Robert Manning, Marie Piasecki, Norman Prokop, and Dr. Daniel Raible for their contribution of time, resources, and expertise toward the development and engineering process, as well as their mentorship of the students involved.

The authors also wish to thank Dave Snyder for his support of the launch and recovery process, as well as his assistance with interpreting and modeling altitude based on the pressure readings recorded during flight.

REFERENCES

[1] K. Ashton, “That Internet of Things thing,” RFiD J., vol. 22, no. 7, pp. 97–114, 2009.

[2] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, “Fog computing and its role in the Internet of things,” in Proc. 1st Edition MCC Workshop Mobile Cloud Comput., Helsinki, Finland, 2012, pp. 13–16..

[3] N. J. Cotton and B. M. Wilamowski, "Compensation of Nonlinearities Using Neural Networks Implemented on Inexpensive Microcontrollers," in *IEEE Transactions on Industrial Electronics*, vol. 58, no. 3, pp. 733-740, March 2011.

[4] F. Akopyan *et al*., "TrueNorth: Design and Tool Flow of a 65 mW 1 Million Neuron Programmable Neurosynaptic Chip," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 10, pp. 1537-1557, Oct. 2015.

[5] L. Cavigelli *et al*. “Origami: A Convolutional Network Accelerator,” in *Proceedings of the 25th edition on Great Lakes Symposium on VLSI (GLSVLSI '15)*. ACM, New York, NY, USA, 2015, 199-204.

[6] Nagasubramanian, G. “Electrical characteristics of 18650 Li-ion cells at low temperatures,” *in Journal of Applied Electrochemistry (2001)* 31: 99. https://doi.org/10.1023/A:1004113825283

[7] Santoni, F., Tortora, P., Alessandrini, F. Passerini, S. “Commercial Li-Ion Batteries for Nanosatellite Applications - a Flight Experiment.” (2002). 502. 653.

[8] V. Vanhoucke, A. Senior, and M. Z. Mao, “Improving the speed of neural networks on CPUs,” in Proc. Deep Learning and Unsupervised Feature Learning NIPS Workshop, 2011 [Online]. Available: http://research.google.com/pubs/ archive/37631.pdf

[9] T. Amert, N. Otterness, M. Yang, J. H. Anderson and F. D. Smith, "GPU Scheduling on the NVIDIA TX2: Hidden Details Revealed," *2017 IEEE Real-Time Systems Symposium (RTSS)*, Paris, 2017, pp. 104-115.

[10] J. Zhang and J. Li. “Improving the Performance of OpenCL-based FPGA Accelerator for Convolutional Neural Network,” in *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA '17)*. ACM, New York, NY, USA, 2017, 25-34.

[11] M. Alawad and M. Lin, "Scalable FPGA Accelerator for Deep Convolutional Neural Networks with Stochastic Streaming," in *IEEE Transactions on Multi-Scale Computing Systems*, vol. 4, no. 4, pp. 888-899, 1 Oct.-Dec. 2018.

[12] D. Moloney, B. Barry, R. Richmond, F. Connor, C. Brick and D. Donohoe, "Myriad 2: Eye of the computational vision storm," *2014 IEEE Hot Chips 26 Symposium (HCS)*, Cupertino, CA, 2014, pp. 1-18.

[13] M. Davies *et al*., "Loihi: A Neuromorphic Manycore Processor with On-Chip Learning," in *IEEE Micro*, vol. 38, no. 1, pp. 82-99, January/February 2018.

[14] J. C. Mankins, "Technology Readiness Assessments: A Retrospective", *Acta Astronautica, 65*, No. 9-10, pp. 1216–1223, 2009.